Summary

Computer Security Lecture 17

Mike Just[1]

School of Informatics
University of Edinburgh

15th March 2010

[1] Based on original lecture notes by David Aspinall

Outline

Programming against security

Techniques for threat analysis

From security evaluation to security management

Revision

Outline

Malware

- Malware or *malicious logic* is code that intentionally sets out to violate a site's security policy.

Malware

- Malware or *malicious logic* is code that intentionally sets out to violate a site's security policy.
- A **Trojan horse** is a program with an overt effect and a covert (usually malicious) effect. Trojans may be self-replicating.

Malware

- Malware or *malicious logic* is code that intentionally sets out to violate a site's security policy.
- A **Trojan horse** is a program with an overt effect and a covert (usually malicious) effect. Trojans may be self-replicating.
- A **virus** is a program that inserts itself into other files and then (usually) performs some action. Two phases: *insertion* and *execution*. Identified by Fred Cohen when a graduate student in 1983, named by his teacher Adleman.

Malware

- Malware or *malicious logic* is code that intentionally sets out to violate a site's security policy.
- A **Trojan horse** is a program with an overt effect and a covert (usually malicious) effect. Trojans may be self-replicating.
- A **virus** is a program that inserts itself into other files and then (usually) performs some action. Two phases: *insertion* and *execution*. Identified by Fred Cohen when a graduate student in 1983, named by his teacher Adleman.
  - Many types of viruses (antivirus companies enjoy categorizations): boot sector, application, TSR, multipartite, encrypted, polymorphic (designed to resist detection), macro.

Malware

- Malware or *malicious logic* is code that intentionally sets out to violate a site's security policy.
- A **Trojan horse** is a program with an overt effect and a covert (usually malicious) effect. Trojans may be self-replicating.
- A **virus** is a program that inserts itself into other files and then (usually) performs some action. Two phases: *insertion* and *execution*. Identified by Fred Cohen when a graduate student in 1983, named by his teacher Adleman.
  - Many types of viruses (antivirus companies enjoy categorizations): boot sector, application, TSR, multipartite, encrypted, polymorphic (designed to resist detection), macro.
- A **worm** is a program that copies itself from one machine to another. Research began on benign worms for distributed computation (current term: "agents").

Crimeware

- Subcategory of malware supporting criminal activity. E.g., identity theft, extortion, fraud, corporate espionage, surveillance, ransomware.

Crimeware

- Subcategory of malware supporting criminal activity. E.g., identity theft, extortion, fraud, corporate espionage, surveillance, ransomware.
- **Keyloggers** and **screenloggers** install in web browser or device driver, send data to phishing server.

Crimeware

- Subcategory of malware supporting criminal activity. E.g., identity theft, extortion, fraud, corporate espionage, surveillance, ransomware.
- **Keyloggers** and **screenloggers** install in web browser or device driver, send data to phishing server.
- A **web Trojan** emulates a web login screen or popup, to steal credentials.

Crimeware

- Subcategory of malware supporting criminal activity. E.g., identity theft, extortion, fraud, corporate espionage, surveillance, ransomware.
- **Keyloggers** and **screenloggers** install in web browser or device driver, send data to phishing server.
- A **web Trojan** emulates a web login screen or popup, to steal credentials.
- A **rootkit** is malware that conceals its own presence, usually subverting system tools (e.g., ps or kernel patch). Virtualization can help this!

Crimeware

- Subcategory of malware supporting criminal activity. E.g., identity theft, extortion, fraud, corporate espionage, surveillance, ransomware.
- **Keyloggers** and **screenloggers** install in web browser or device driver, send data to phishing server.
- A **web Trojan** emulates a web login screen or popup, to steal credentials.
- A **rootkit** is malware that conceals its own presence, usually subverting system tools (e.g., ps or kernel patch). Virtualization can help this!
- Distribution: piggy-backing, worms, browser exploits, manual cracking, affiliate marketing.

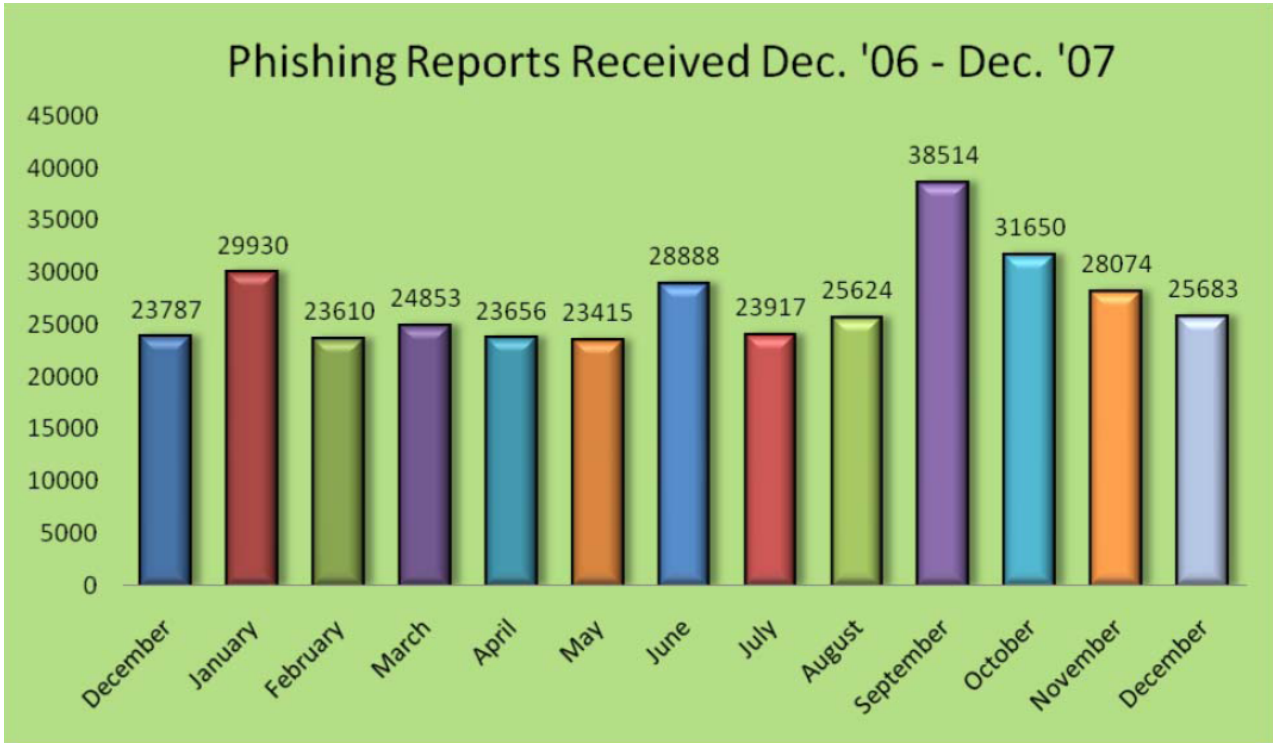
Crimeware

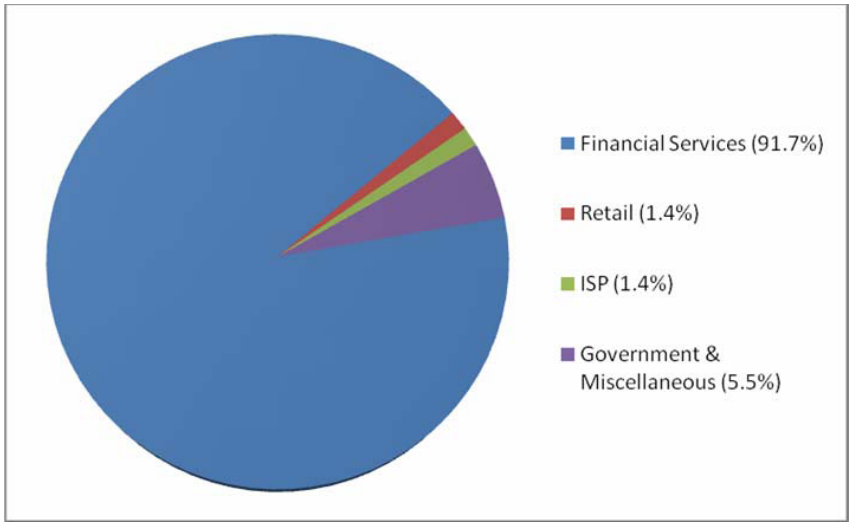
- Subcategory of malware supporting criminal activity. E.g., identity theft, extortion, fraud, corporate espionage, surveillance, ransomware.
- **Keyloggers** and **screenloggers** install in web browser or device driver, send data to phishing server.
- A **web Trojan** emulates a web login screen or popup, to steal credentials.
- A **rootkit** is malware that conceals its own presence, usually subverting system tools (e.g., ps or kernel patch). Virtualization can help this!
- Distribution: piggy-backing, worms, browser exploits, manual cracking, affiliate marketing.
- Countermeasures: good keyloggers, information leakage detectors; others described earlier.

Crimeware

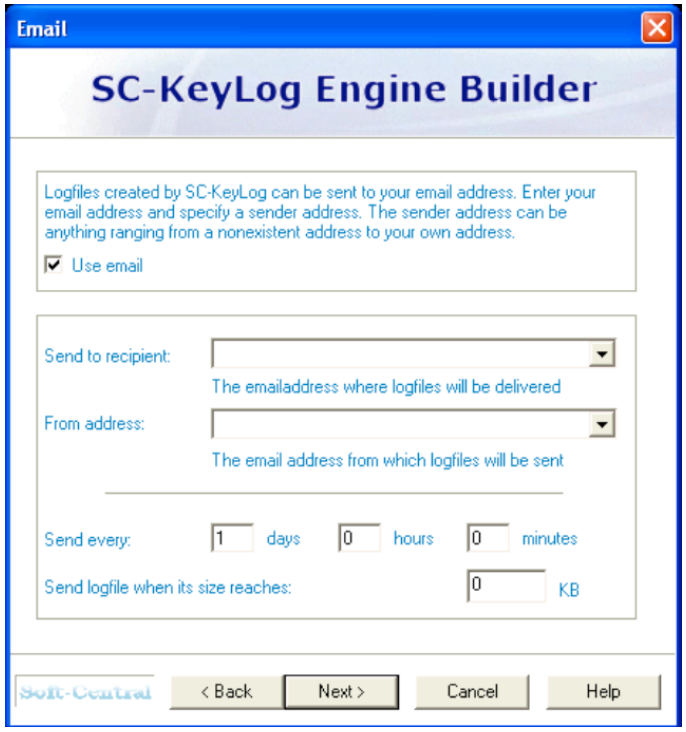
- Subcategory of malware supporting criminal activity. E.g., identity theft, extortion, fraud, corporate espionage, surveillance, ransomware.
- **Keyloggers** and **screenloggers** install in web browser or device driver, send data to phishing server.
- A **web Trojan** emulates a web login screen or popup, to steal credentials.
- A **rootkit** is malware that conceals its own presence, usually subverting system tools (e.g., `ps` or kernel patch). Virtualization can help this!
- Distribution: piggy-backing, worms, browser exploits, manual cracking, affiliate marketing.
- Countermeasures: good keyloggers, information leakage detectors; others described earlier.
- For more, see `http://www.antiphishing.org/`. Following images are from APWG reports.

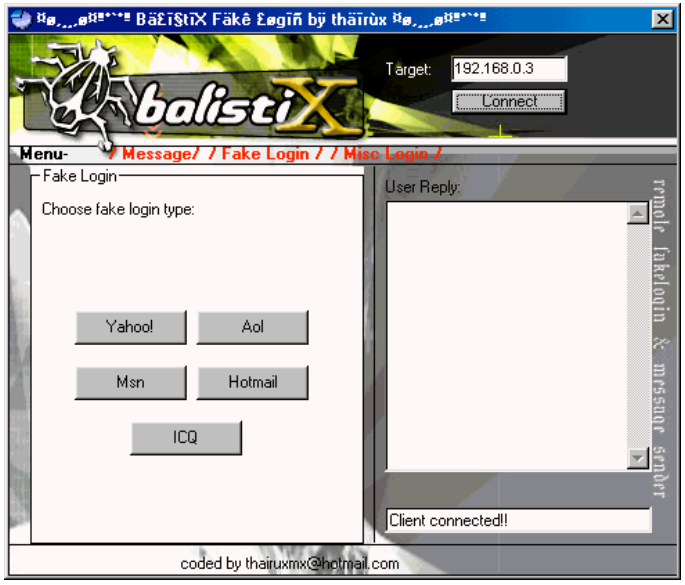
Phishing

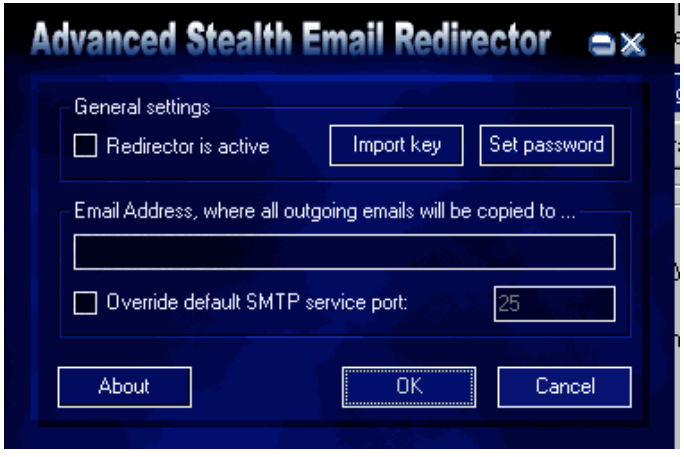

Phishing Reports Received Dec. '06 - Dec. '07
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
23787
29930
23610
24853
23656
23415
28888
23917
25624
38514
31650
28074
25683
December
January
February
March
April
May
June
July
August
September
October
November
December


Financial Services (91.7%)
Retail (1.4%)
ISP (1.4%)
Government & Miscellaneous (5.5%)

Building crimeware







Outline

Attack Trees

- Use attack data & known vulnerabilities to identify possible attacks against a system. Special case of *fault tree analysis*.

Attack Trees

- Use attack data & known vulnerabilities to identify possible attacks against a system. Special case of *fault tree analysis*.
- Formally: for each attack target, give a tree with AND-nodes and OR-nodes which contain subgoals necessary to achieve attack. Successive refinement with subtrees gives greater detail in how attack is achieved. Traversal of tree gives possible *attack scenarios* as sequences of conditions in leaves.

Attack Trees

- Use attack data & known vulnerabilities to identify possible attacks against a system. Special case of *fault tree analysis*.
- Formally: for each attack target, give a tree with AND-nodes and OR-nodes which contain subgoals necessary to achieve attack. Successive refinement with subtrees gives greater detail in how attack is achieved. Traversal of tree gives possible *attack scenarios* as sequences of conditions in leaves.
- Nodes can be labelled with probabilities or *possibility/impossibility* flags, or by calculating from leaves, cost estimates of attack. Trees may be annotated with possible countermeasures.

Attack Trees

- Use attack data & known vulnerabilities to identify possible attacks against a system. Special case of *fault tree analysis*.
- Formally: for each attack target, give a tree with AND-nodes and OR-nodes which contain subgoals necessary to achieve attack. Successive refinement with subtrees gives greater detail in how attack is achieved. Traversal of tree gives possible *attack scenarios* as sequences of conditions in leaves.
- Nodes can be labelled with probabilities or *possibility/impossibility* flags, or by calculating from leaves, cost estimates of attack. Trees may be annotated with possible countermeasures.
- Hope: security analyst can combine and reuse existing attack patterns to help security evaluation.

Attack Trees

- Use attack data & known vulnerabilities to identify possible attacks against a system. Special case of *fault tree analysis*.
- Formally: for each attack target, give a tree with AND-nodes and OR-nodes which contain subgoals necessary to achieve attack. Successive refinement with subtrees gives greater detail in how attack is achieved. Traversal of tree gives possible *attack scenarios* as sequences of conditions in leaves.
- Nodes can be labelled with probabilities or *possibility/impossibility* flags, or by calculating from leaves, cost estimates of attack. Trees may be annotated with possible countermeasures.
- Hope: security analyst can combine and reuse existing attack patterns to help security evaluation.
- Obvious drawback:

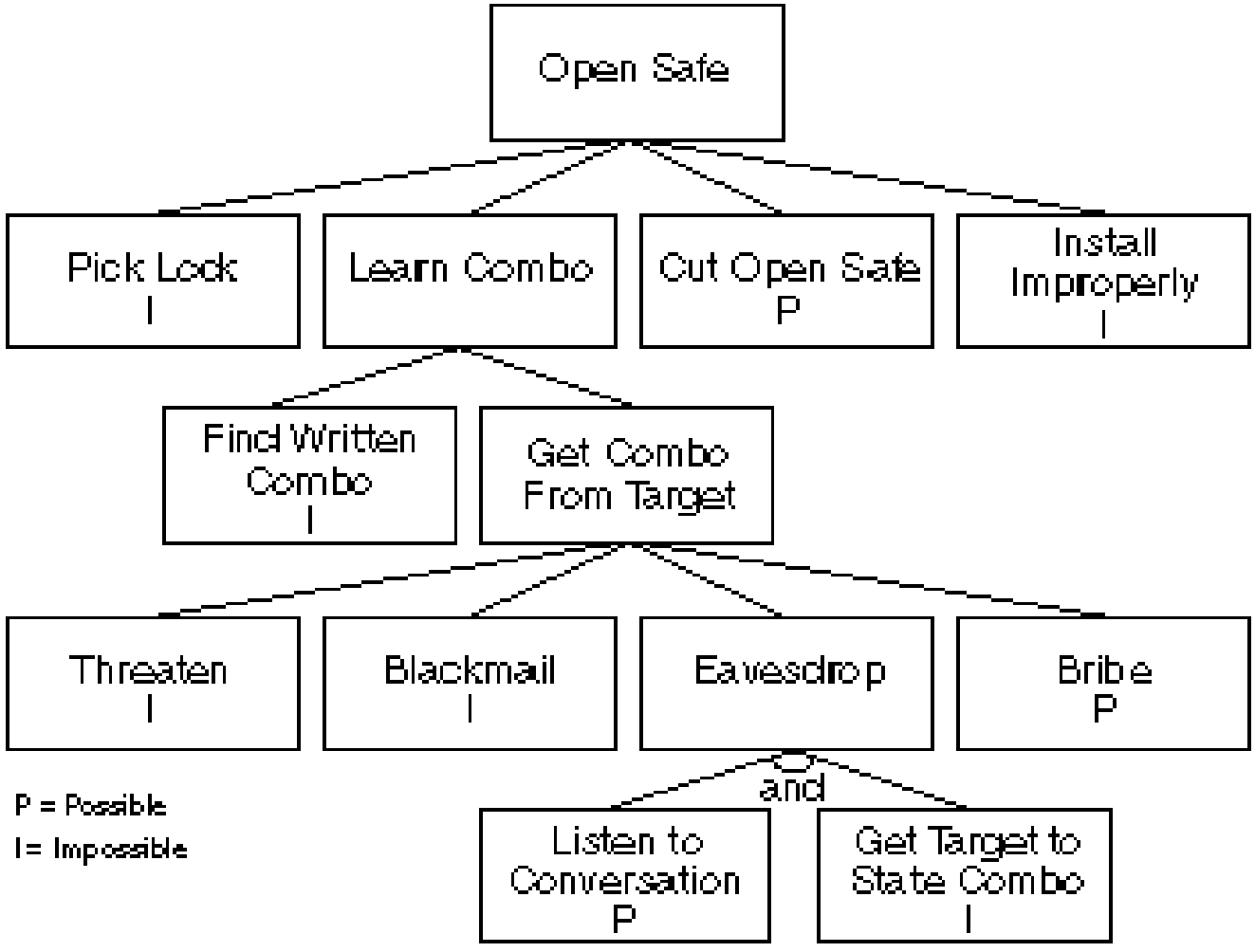
Attack Trees

- Use attack data & known vulnerabilities to identify possible attacks against a system. Special case of *fault tree analysis*.
- Formally: for each attack target, give a tree with AND-nodes and OR-nodes which contain subgoals necessary to achieve attack. Successive refinement with subtrees gives greater detail in how attack is achieved. Traversal of tree gives possible *attack scenarios* as sequences of conditions in leaves.
- Nodes can be labelled with probabilities or *possibility/impossibility* flags, or by calculating from leaves, cost estimates of attack. Trees may be annotated with possible countermeasures.
- Hope: security analyst can combine and reuse existing attack patterns to help security evaluation.
- Obvious drawback:

Attack Trees

- Use attack data & known vulnerabilities to identify possible attacks against a system. Special case of *fault tree analysis*.
- Formally: for each attack target, give a tree with AND-nodes and OR-nodes which contain subgoals necessary to achieve attack. Successive refinement with subtrees gives greater detail in how attack is achieved. Traversal of tree gives possible *attack scenarios* as sequences of conditions in leaves.
- Nodes can be labelled with probabilities or *possibility/impossibility* flags, or by calculating from leaves, cost estimates of attack. Trees may be annotated with possible countermeasures.
- Hope: security analyst can combine and reuse existing attack patterns to help security evaluation.
- Obvious drawback:only models known/pre-existing attacks; biggest threats may come from new, unknown attacks.

Attack Tree for Safe Cracking [Sch99]



Attack scenarios generated by depth-first tree traversals excluding impossible cases.

Attack Tree for ACME Web server [MEL01]

Access sensitive data from privileged account at ACME

AND 1. Get access to privileged account on web server
- *OR* 1. Exploit buffer overflow vulnerability to access privileged account
  - *AND* 1. Identify executable program on ACME Web server susceptible to buffer overflow vulnerability
    2. Identify code that would provide access . . .
  - 2. Exploit unexpected operator vulnerability to access privileged account
  - *AND* 1. Find executable program on ACME Web server susceptible to vulnerability
    2. Identify (unexpected) operator that permits composing system calls
    3. Identify system call that would provide access to privileged account . . .

2. Scan files for sensitive data

STRIDE [HL03]

- A mnemonic for remembering categories of software threats.

STRIDE

Spoofing e.g., attacker pretends to be someone else

STRIDE [HL03]

- A mnemonic for remembering categories of software threats.

STRIDE

Spoofing e.g., attacker pretends to be someone else

Tampering e.g., attacker alters data or settings

STRIDE [HL03]

- A mnemonic for remembering categories of software threats.

STRIDE

Spoofing e.g., attacker pretends to be someone else

Tampering e.g., attacker alters data or settings

Repudiation e.g., user denies making attack, spending money

STRIDE [HL03]

- A mnemonic for remembering categories of software threats.

STRIDE

Spoofing e.g., attacker pretends to be someone else

Tampering e.g., attacker alters data or settings

Repudiation e.g., user denies making attack, spending money

Information disclosure e.g., loss of personal information

STRIDE [HL03]

- A mnemonic for remembering categories of software threats.

STRIDE

Spoofing e.g., attacker pretends to be someone else

Tampering e.g., attacker alters data or settings

Repudiation e.g., user denies making attack, spending money

Information disclosure e.g., loss of personal information

Denial of service e.g., preventing e-commerce site operation

STRIDE [HL03]

- A mnemonic for remembering categories of software threats.

STRIDE

Spoofing e.g., attacker pretends to be someone else

Tampering e.g., attacker alters data or settings

Repudiation e.g., user denies making attack, spending money

Information disclosure e.g., loss of personal information

Denial of service e.g., preventing e-commerce site operation

Elevation of privilege e.g., user illegitimately gains power of root user

DREAD [HL03]

- A mnemonic for issues that might influence a risk assessment for a software threat.

DREAD [HL03]

- A mnemonic for issues that might influence a risk assessment for a software threat.

DREAD

Damage potential value lost for assets affected

DREAD [HL03]

- A mnemonic for issues that might influence a risk assessment for a software threat.

DREAD

Damage potential value lost for assets affected

Reproducibility how easy it is to realise the threat

DREAD [HL03]

- A mnemonic for issues that might influence a risk assessment for a software threat.

DREAD

Damage potential	value lost for assets affected
Reproducibility	how easy it is to realise the threat
Exploitability	expertise and resources needed for attack

DREAD [HL03]

- A mnemonic for issues that might influence a risk assessment for a software threat.

DREAD

Damage potential value lost for assets affected

Reproducibility how easy it is to realise the threat

Exploitability expertise and resources needed for attack

Affected Users how many users affected (for a multi-user system)

DREAD [HL03]

- A mnemonic for issues that might influence a risk assessment for a software threat.

DREAD

Damage potential	value lost for assets affected
Reproducibility	how easy it is to realise the threat
Exploitability	expertise and resources needed for attack
Affected Users	how many users affected (for a multi-user system)
Discoverability	likelihood of detecting the attack

DREAD [HL03]

- A mnemonic for issues that might influence a risk assessment for a software threat.

DREAD

Damage potential	value lost for assets affected
Reproducibility	how easy it is to realise the threat
Exploitability	expertise and resources needed for attack
Affected Users	how many users affected (for a multi-user system)
Discoverability	likelihood of detecting the attack

DREAD [HL03]

- A mnemonic for issues that might influence a risk assessment for a software threat.

DREAD	
Damage potential	value lost for assets affected
Reproducibility	how easy it is to realise the threat
Exploitability	expertise and resources needed for attack
Affected Users	how many users affected (for a multi-user system)
Discoverability	likelihood of detecting the attack

- In terms of risk assessment, Damage and Affected Users are measures of **impact**, reproducibility, exploitability and discoverability are measures of **likelihood**.

Outline

Reminder: security properties

In computer (or information) security, we are concerned with ensuring certain *security properties* of our assets:

Security Properties to Ensure

confidentiality *improper information gathering prevented*

Reminder: security properties

In computer (or information) security, we are concerned with ensuring certain *security properties* of our assets:

Security Properties to Ensure

confidentiality	*improper information gathering prevented*
integrity	*data has not been (maliciously) altered*

Reminder: security properties

In computer (or information) security, we are concerned with ensuring certain *security properties* of our assets:

Security Properties to Ensure

confidentiality	*improper information gathering prevented*
integrity	*data has not been (maliciously) altered*
availability	*data/services can be accessed as desired*

Reminder: security properties

In computer (or information) security, we are concerned with ensuring certain *security properties* of our assets:

Security Properties to Ensure

confidentiality	*improper information gathering prevented*
integrity	*data has not been (maliciously) altered*
availability	*data/services can be accessed as desired*
accountability	*actions traceable to those responsible*

Reminder: security properties

In computer (or information) security, we are concerned with ensuring certain *security properties* of our assets:

Security Properties to Ensure

confidentiality	*improper information gathering prevented*
integrity	*data has not been (maliciously) altered*
availability	*data/services can be accessed as desired*
accountability	*actions traceable to those responsible*
authentication	*user or data origin accurately identifiable*

Reminder: security properties

In computer (or information) security, we are concerned with ensuring certain *security properties* of our assets:

Security Properties to Ensure

confidentiality	*improper information gathering prevented*
integrity	*data has not been (maliciously) altered*
availability	*data/services can be accessed as desired*
accountability	*actions traceable to those responsible*
authentication	*user or data origin accurately identifiable*

In most systems we want to protect all of these properties, in various specific ways. Different mechanisms may be used to provide protection, but from the start one should realise that **security protection is a whole-system issue**. Moreover, the whole system may be used in the most inclusive sense.

Reminder: security properties

In computer (or information) security, we are concerned with ensuring certain *security properties* of our assets:

Security Properties to Ensure	
confidentiality	*improper information gathering prevented*
integrity	*data has not been (maliciously) altered*
availability	*data/services can be accessed as desired*
accountability	*actions traceable to those responsible*
authentication	*user or data origin accurately identifiable*

In most systems we want to protect all of these properties, in various specific ways. Different mechanisms may be used to provide protection, but from the start one should realise that **security protection is a whole-system issue**. Moreover, the whole system may be used in the most inclusive sense. Security evaluation standards for products are restrictive here, but **security management standards** attempt to cover the whole picture.

Security Management ISO 27001/2

- ISO 27001/2 (formerly ISO17799) describes how an organization can set up an *Information Security Management System* (ISMS), focusing on operational procedures as well as technical security controls.

Security Management ISO 27001/2

- ISO 27001/2 (formerly ISO17799) describes how an organization can set up an *Information Security Management System* (ISMS), focusing on operational procedures as well as technical security controls.
- Information security is defined as preservation of confidentiality, integrity and availability of information.

Security Management ISO 27001/2

- ISO 27001/2 (formerly ISO17799) describes how an organization can set up an *Information Security Management System* (ISMS), focusing on operational procedures as well as technical security controls.
- Information security is defined as preservation of confidentiality, integrity and availability of information.
- Among other things, the ISMS will define:

Security Management ISO 27001/2

- ISO 27001/2 (formerly ISO17799) describes how an organization can set up an *Information Security Management System* (ISMS), focusing on operational procedures as well as technical security controls.
- Information security is defined as preservation of confidentiality, integrity and availability of information.
- Among other things, the ISMS will define:
  - A *scope* in terms of the organization and its assets;

Security Management ISO 27001/2

- ISO 27001/2 (formerly ISO17799) describes how an organization can set up an *Information Security Management System* (ISMS), focusing on operational procedures as well as technical security controls.
- Information security is defined as preservation of confidentiality, integrity and availability of information.
- Among other things, the ISMS will define:
  - A *scope* in terms of the organization and its assets;
  - A *policy* which sets security objectives; considers legal, regulatory and contractual requirements; sets a context and criteria for risk assessment and management (whether risks are acceptable or should be controlled, avoided, or transferred).

Security Management ISO 27001/2

- ISO 27001/2 (formerly ISO17799) describes how an organization can set up an *Information Security Management System* (ISMS), focusing on operational procedures as well as technical security controls.
- Information security is defined as preservation of confidentiality, integrity and availability of information.
- Among other things, the ISMS will define:
  - A *scope* in terms of the organization and its assets;
  - A *policy* which sets security objectives; considers legal, regulatory and contractual requirements; sets a context and criteria for risk assessment and management (whether risks are acceptable or should be controlled, avoided, or transferred).
- Achieving compliance with the processes required in the standard is a significant undertaking for an organisation (cf ISO 9000).

Sample of ISO 27001/2 areas

4. **Organizational security**: management forum; responsibility allocation; independent reviews; 3rd party and outsourcing

Sample of ISO 27001/2 areas

4. **Organizational security**: management forum; responsibility allocation; independent reviews; 3rd party and outsourcing
5. **Asset classification**: inventory; data labelling

Sample of ISO 27001/2 areas

4. **Organizational security**: management forum; responsibility allocation; independent reviews; 3rd party and outsourcing
5. **Asset classification**: inventory; data labelling
6. **Personnel security**: security in job descriptions and contracts; training; response mechanisms; disciplinary procedures.

Sample of ISO 27001/2 areas

4. **Organizational security**: management forum; responsibility allocation; independent reviews; 3rd party and outsourcing
5. **Asset classification**: inventory; data labelling
6. **Personnel security**: security in job descriptions and contracts; training; response mechanisms; disciplinary procedures.
7. **Environmental security**: physical perimeters; entry controls; delivery areas; equipment siting, power, cabling, disposal; clear desk and screen policy; property removal rules.

Sample of ISO 27001/2 areas

4. **Organizational security**: management forum; responsibility allocation; independent reviews; 3rd party and outsourcing
5. **Asset classification**: inventory; data labelling
6. **Personnel security**: security in job descriptions and contracts; training; response mechanisms; disciplinary procedures.
7. **Environmental security**: physical perimeters; entry controls; delivery areas; equipment siting, power, cabling, disposal; clear desk and screen policy; property removal rules.
8. **Operations management**: documented procedures; change control; segregation of duties; separation of development and operational facilities; malware controls; backups and logs; network management; media handling; information exchange email, agreements, e-commerce, . . .

Sample of ISO 27001/2 areas, cont’d

9. **Access control**: policy specification; user management; network controls; OS and application controls; monitoring; mobile and teleworking.

Sample of ISO 27001/2 areas, cont'd

9. **Access control**: policy specification; user management; network controls; OS and application controls; monitoring; mobile and teleworking.
10. **Systems development**: requirements analysis; authentication, input/processing/output validation; cryptographic controls; system file security; security in development processes: version control, OS/app changes, outsourcing.

Sample of ISO 27001/2 areas, cont'd

9. **Access control**: policy specification; user management; network controls; OS and application controls; monitoring; mobile and teleworking.
10. **Systems development**: requirements analysis; authentication, input/processing/output validation; cryptographic controls; system file security; security in development processes: version control, OS/app changes, outsourcing.
11. **Business continuity**: contingency plans for timely recovery after disasters, security failures, loss of service.

Sample of ISO 27001/2 areas, cont'd

9. **Access control**: policy specification; user management; network controls; OS and application controls; monitoring; mobile and teleworking.
10. **Systems development**: requirements analysis; authentication, input/processing/output validation; cryptographic controls; system file security; security in development processes: version control, OS/app changes, outsourcing.
11. **Business continuity**: contingency plans for timely recovery after disasters, security failures, loss of service.
12. **Compliance** with legal requirements (IPR, DP, copyright, cryptography use, evidence collection, . . .); systems compliance; audit protection.

Failure categorizations

- **Failure in design**

Failure categorizations

- **Failure in design**
  - cryptographic protocol open to replay attack

Failure categorizations

- **Failure in design**
  - cryptographic protocol open to replay attack
  - document format allows code execution (arguable)

Failure categorizations

- **Failure in design**
  - cryptographic protocol open to replay attack
  - document format allows code execution (arguable)
- **Failure in implementation**

Failure categorizations

- **Failure in design**
  - cryptographic protocol open to replay attack
  - document format allows code execution (arguable)
- **Failure in implementation**
  - cryptographically strong PRNG uses predictable seed

Failure categorizations

- **Failure in design**
  - cryptographic protocol open to replay attack
  - document format allows code execution (arguable)
- **Failure in implementation**
  - cryptographically strong PRNG uses predictable seed
  - web application vulnerable to injection attack

Failure categorizations

- **Failure in design**
  - cryptographic protocol open to replay attack
  - document format allows code execution (arguable)
- **Failure in implementation**
  - cryptographically strong PRNG uses predictable seed
  - web application vulnerable to injection attack
- **Failure in operation**

Failure categorizations

- **Failure in design**
  - cryptographic protocol open to replay attack
  - document format allows code execution (arguable)
- **Failure in implementation**
  - cryptographically strong PRNG uses predictable seed
  - web application vulnerable to injection attack
- **Failure in operation**
  - user writes password on PostIt note

Failure categorizations

- **Failure in design**
  - cryptographic protocol open to replay attack
  - document format allows code execution (arguable)
- **Failure in implementation**
  - cryptographically strong PRNG uses predictable seed
  - web application vulnerable to injection attack
- **Failure in operation**
  - user writes password on PostIt note
  - password accidentally typed into name field

Outline

Reminder: Protocols

- Careful protocol design helps establish and maintain security properties. We saw *authentication* and *key-exchange* protocols.

Reminder: Protocols

- Careful protocol design helps establish and maintain security properties. We saw *authentication* and *key-exchange* protocols.
- Shared key authentication, vulnerable to replay:

$$A \rightarrow S : \quad A, \{ A \}_{K_{as}}$$

Reminder: Protocols

- Careful protocol design helps establish and maintain security properties. We saw *authentication* and *key-exchange* protocols.
- Shared key authentication, vulnerable to replay:

$$A \rightarrow S : \quad A, \{ A \}_{K_{as}}$$

- Challenge-response with *fresh* component:

$$\begin{array}{lll} \text{Message 1.} & S \rightarrow A: & N \\ \text{Message 2.} & A \rightarrow S: & \{ A, N \}_{K_{as}} \end{array}$$

Reminder: Protocols

- Careful protocol design helps establish and maintain security properties. We saw *authentication* and *key-exchange* protocols.
- Shared key authentication, vulnerable to replay:

$$A \rightarrow S: \quad A, \{A\}_{K_{as}}$$

- Challenge-response with *fresh* component:

$$\begin{array}{lll} \text{Message 1.} & S \rightarrow A: & N \\ \text{Message 2.} & A \rightarrow S: & \{A, N\}_{K_{as}} \end{array}$$

- Mutual authentication with shared keys:

$$\begin{array}{lll} \text{Message 1.} & S \rightarrow A: & N_s \\ \text{Message 2.} & A \rightarrow S: & \{N_s, N_a, S\}_{K_{as}} \\ \text{Message 3.} & S \rightarrow A: & \{N_a, N_s\}_{K_{as}} \end{array}$$

Reminder: Cryptography

- We considered **cryptographic primitives** based on 0, 1 or 2 keys: hash functions; symmetric-key ciphers; public-key ciphers and digital signatures.

Reminder: Cryptography

- We considered **cryptographic primitives** based on 0, 1 or 2 keys: hash functions; symmetric-key ciphers; public-key ciphers and digital signatures.
- A cipher (symmetric or asymmetric) is defined by giving $(\mathcal{M}, \mathcal{C}, \mathcal{K}, E_e, D_d)$ of: *message space*, *cipher space*, *key space*, *encryption function* and *decryption function*.

Reminder: Cryptography

- We considered **cryptographic primitives** based on 0, 1 or 2 keys: hash functions; symmetric-key ciphers; public-key ciphers and digital signatures.
- A cipher (symmetric or asymmetric) is defined by giving $(\mathcal{M}, \mathcal{C}, \mathcal{K}, E_e, D_d)$ of: *message space*, *cipher space*, *key space*, *encryption function* and *decryption function*.
- Symmetric constructions: stream (one-time pad and LFSR) and block (substitution, permutation, product)

Reminder: Cryptography

- We considered **cryptographic primitives** based on 0, 1 or 2 keys: hash functions; symmetric-key ciphers; public-key ciphers and digital signatures.
- A cipher (symmetric or asymmetric) is defined by giving $(\mathcal{M}, \mathcal{C}, \mathcal{K}, E_e, D_d)$ of: *message space*, *cipher space*, *key space*, *encryption function* and *decryption function*.
- Symmetric constructions: stream (one-time pad and LFSR) and block (substitution, permutation, product)
- Hash function properties (OWHFs and CRHFs). Uses: fingerprints, signatures, knowledge confirmation/commitment, key derivation, PRNGs.

Reminder: Cryptography

- We considered **cryptographic primitives** based on 0, 1 or 2 keys: hash functions; symmetric-key ciphers; public-key ciphers and digital signatures.
- A cipher (symmetric or asymmetric) is defined by giving $(\mathcal{M}, \mathcal{C}, \mathcal{K}, E_e, D_d)$ of: *message space*, *cipher space*, *key space*, *encryption function* and *decryption function*.
- Symmetric constructions: stream (one-time pad and LFSR) and block (substitution, permutation, product)
- Hash function properties (OWHFs and CRHFs). Uses: fingerprints, signatures, knowledge confirmation/commitment, key derivation, PRNGs.
- Asymmetric ciphers: RSA, Diffie-Hellman, ElGamal

Reminder: Network security

- Attacks, including:

Reminder: Network security

- Attacks, including:
  - SYN flooding

Reminder: Network security

- Attacks, including:
  - SYN flooding
  - smurfing

Reminder: Network security

- Attacks, including:
  - SYN flooding
  - smurfing
  - DNS

Reminder: Network security

- Attacks, including:
  - SYN flooding
  - smurfing
  - DNS
  - sequence numbers

Reminder: Network security

- Attacks, including:
  - SYN flooding
  - smurfing
  - DNS
  - sequence numbers
- Vulnerable protocols: UDP, RPC, NFS, NIS, X-Window, SNMP

Reminder: Network security

- Attacks, including:
  - SYN flooding
  - smurfing
  - DNS
  - sequence numbers
- Vulnerable protocols: UDP, RPC, NFS, NIS, X-Window, SNMP
- Defences, including:

Reminder: Network security

- Attacks, including:
  - SYN flooding
  - smurfing
  - DNS
  - sequence numbers
- Vulnerable protocols: UDP, RPC, NFS, NIS, X-Window, SNMP
- Defences, including:
  - Firewalls: packet filters, application gateways, circuit relays

Reminder: Network security

- Attacks, including:
  - SYN flooding
  - smurfing
  - DNS
  - sequence numbers
- Vulnerable protocols: UDP, RPC, NFS, NIS, X-Window, SNMP
- Defences, including:
  - Firewalls: packet filters, application gateways, circuit relays
  - Firewall issues: configuration, maintenance, tunnelling

Reminder: Network security

- Attacks, including:
  - SYN flooding
  - smurfing
  - DNS
  - sequence numbers
- Vulnerable protocols: UDP, RPC, NFS, NIS, X-Window, SNMP
- Defences, including:
  - Firewalls: packet filters, application gateways, circuit relays
  - Firewall issues: configuration, maintenance, tunnelling
  - Logging, auditing, forensics

Reminder: Network security

- Attacks, including:
  - SYN flooding
  - smurfing
  - DNS
  - sequence numbers
- Vulnerable protocols: UDP, RPC, NFS, NIS, X-Window, SNMP
- Defences, including:
  - Firewalls: packet filters, application gateways, circuit relays
  - Firewall issues: configuration, maintenance, tunnelling
  - Logging, auditing, forensics
  - Intrusion detection

Reminder: Network security

- Attacks, including:
  - SYN flooding
  - smurfing
  - DNS
  - sequence numbers
- Vulnerable protocols: UDP, RPC, NFS, NIS, X-Window, SNMP
- Defences, including:
  - Firewalls: packet filters, application gateways, circuit relays
  - Firewall issues: configuration, maintenance, tunnelling
  - Logging, auditing, forensics
  - Intrusion detection
  - Honeypots

Reminder: Network security

- Attacks, including:
  - SYN flooding
  - smurfing
  - DNS
  - sequence numbers
- Vulnerable protocols: UDP, RPC, NFS, NIS, X-Window, SNMP
- Defences, including:
  - Firewalls: packet filters, application gateways, circuit relays
  - Firewall issues: configuration, maintenance, tunnelling
  - Logging, auditing, forensics
  - Intrusion detection
  - Honeypots
- Adding security layers: link-level, network-level and application-level.

Reminder: Network security

- Attacks, including:
  - SYN flooding
  - smurfing
  - DNS
  - sequence numbers
- Vulnerable protocols: UDP, RPC, NFS, NIS, X-Window, SNMP
- Defences, including:
  - Firewalls: packet filters, application gateways, circuit relays
  - Firewall issues: configuration, maintenance, tunnelling
  - Logging, auditing, forensics
  - Intrusion detection
  - Honeypots
- Adding security layers: link-level, network-level and application-level.
- In overview: IPsec, DNSSec, SSH, VPNs

Reminder: Security Models

- **Security models**: frameworks for specifying security policies, typically enforcing access control or information flow.

Reminder: Security Models

- **Security models**: frameworks for specifying security policies, typically enforcing access control or information flow.
- A policy may have a *discretionary control* (DAC) aspect (user-defined) and a *mandatory control* (MAC) aspect (system-wide) which takes precedence.

Reminder: Security Models

- **Security models**: frameworks for specifying security policies, typically enforcing access control or information flow.
- A policy may have a *discretionary control* (DAC) aspect (user-defined) and a *mandatory control* (MAC) aspect (system-wide) which takes precedence.
- DAC is defined by an access control matrix (represented by row or by column); MAC is defined by *policy rules*, typically referring to **security levels** or **domains and types**.

Reminder: Security Models

- **Security models**: frameworks for specifying security policies, typically enforcing access control or information flow.
- A policy may have a *discretionary control* (DAC) aspect (user-defined) and a *mandatory control* (MAC) aspect (system-wide) which takes precedence.
- DAC is defined by an access control matrix (represented by row or by column); MAC is defined by *policy rules*, typically referring to **security levels** or **domains and types**.
- State based models have a notion of a *secure state* and a theorem stating that starting from a secure state, the only reachable states are also secure.

Reminder: Security Models

- **Security models**: frameworks for specifying security policies, typically enforcing access control or information flow.
- A policy may have a *discretionary control* (DAC) aspect (user-defined) and a *mandatory control* (MAC) aspect (system-wide) which takes precedence.
- DAC is defined by an access control matrix (represented by row or by column); MAC is defined by *policy rules*, typically referring to **security levels** or **domains and types**.
- State based models have a notion of a *secure state* and a theorem stating that starting from a secure state, the only reachable states are also secure.
- Particular model: Bell-LaPadula.

Reminder: Secure programming

splitvt, syslog, mount/umount, sendmail, lpr, bind, gethostbyname(), modstat, cron, login, sendmail again, the query CGI script, newgrp, AutoSofts RTS inventory control system, host, talkd, getopt(), sendmail yet again, FreeBSD s crt0.c, WebSite 1.1, rlogin, term, ffbconfig, libX11, passwd/yppasswd/nispasswd, imapd, ipop3d, SuperProbe, lpd, xterm, eject, lpd again, host, mount, the NLS library, xlock, libXt and further X11R6 libraries, talkd, fdformat, eject, elm, cxterm, ps, fbconfig, metamail, dtterm, df, an entire range of SGI programs, ps again, chkey, libX11, suidperl, libXt again, lquerylv, getopt() again, dtaction, at, libDtSvc, eeprom, lpr yet again, smbmount, xlock yet again, MH-6.83, NIS+, ordist, xlock again, ps again, bash, rdist, login/scheme, libX11 again, sendmail for Windows NT, wm, wwwcount, tgetent(), xdat, termcap, portmir, writesrv, rcp, opengroup, telnetd, rlogin, MSIE, eject, df, statd, at again, rlogin again, rsh, ping, traceroute, Cisco 7xx routers, xscreensaver, passwd, deliver, cidentd, Xserver, the Yapp conferencing server, multiple problems in the Windows95/NT NTFTP client, the Windows War and Serv-U FTP daemon, the Linux dynamic linker, filter (part of elm-2.4), the IMail POP3 server for NT, pset, rpc.nisd, Samba server, ufsrestore, DCE secd, pine, dslip, Real Player, SLMail, socks5, CSM Proxy, imapd (again), Outlook Express, Netscape Mail, mutt, MSIE, Lotus Notes, MSIE again, libauth, login, iwsh, permissions, unfsd, Minicom, nslookup, zpop, dig, WebCam32, smbclient, compress, elvis, lha, bash, jidentd, Tooltalk, ttdbserver, dbadmin, zgv, mountd, pcnfs, Novell Groupwise, mscreen, xterm, Xaw library, Cisco IOS, mutt again, ospf_monitor, sdtcm_convert, Netscape (all versions), mpg123, Xprt, klogd, catdoc, junkbuster, SerialPOP, and rdist

- This is a year’s worth of (reported) buffer overflow vulnerabilities (2000/1).

References

[Sch99] Bruce Schneier. Attack trees.

Dr Dobb's Journal, December 1999. Available at `http://www.schneier.com/paper-attacktrees-ddj-ft.html`.

Information technology — code of practice for information security management, December 2000.

Standard: ISO/IEC 17799 and BSI BS7799.

[MEL01] Andrew P. Moore, Robert J. Ellison, and Richard C. Linger. Attack modeling for information security and survivability.

Technical Report CMU/SEI-2001-TN-001, Software Engineering Institute, Carnegie Mellon University, 2001.

[HL03] M. Howard and D. LeBlanc. *Writing Secure Code*.

Microsoft Press, second edition, 2003.

Recommended Reading

Schneier's attack tree article. Chapter 1 of Gollmann's textbook *Computer Security*.